10161 Abstracts Collection
Decision Procedures in Software, Hardware and Bioware
— Dagstuhl Seminar —

Nikolaj Bjorner[1], Robert Nieuwenhuis[2], Helmut Veith[3] and Andrei Voronkov[4]

[1] Microsoft Research - Redmond, US
nbjorner@microsoft.com
[2] TU of Catalonia - Barcelona, ES
roberto@lsi.upc.edu
[3] TU Wien, AT
veith@forsyte.de
[4] University of Manchester, GB
Andrei@voronkov.com

Abstract. From April 19th, 2010 to April 23rd, 2010, the Dagstuhl Seminar 10161 *Decision Procedures in Soft, Hard and Bio-ware* was held in Schloss Dagstuhl Leibniz Center for Informatics. During the seminar, several participants presented their current research, and ongoing work and open problems were discussed. Abstracts of the presentations given during the seminar as well as links to slides and links to papers behind the presentations and papers produced as a result of the seminar are put together in this paper. The first section describes the seminar topics and goals in general. Links to extended abstracts or full papers are provided, if available.

Keywords. Decision Procedures, Satisfiability Modulo Theories, Software Verification, Dynamic Symbolic Execution, Interpolants, Hardware Verification, Bio-analysis

10161 Executive Summary – Decision Procedures in Software, Hardware and Bioware

A goal of the seminar Decision Procedures in Soft, Hard and Bio-ware was to bring together renowned as well as young aspiring researchers from two groups. The first group formed by researchers who develop both theory and efficient implementations of decision procedures. The second group comprising of researchers from application areas such as program analysis and testing, cryptoanalysis, hardware verification, industrial planning and scheduling, and bioinformatics, who have worked with, and contributed to, high quality decision procedures. The purpose of the seminar was to heighten awareness between tool and theory developers for decision procedures with the array of applications

found in software, hardware and biological systems analysis. The seminar fell in the week of April 19-23, 2010.

In spite of the travel disruptions due to the Icelandic volcano eruption, 27 researchers from 8 countries (Germany, Austria, Italy, France, USA, United Kingdom, Switzerland, and India) arrived and discussed their recent work and future trends. The absence of several attendees from North America meant that the seminar was unable to cover planned tutorials on Bio-analysis and constraint solving. Instead it focused heavily on decision procedures in the context of software analysis and to some extent hardware. The planned tutorial on hardware analysis, by Armin Biere, was highly successful, though. On the other hand, it allowed for a highly interactive environment and some attendees got a chance to present a talk on more than one topic.

The following summarizes the talk abstracts.

Joint work of: Bjorner, Nikolaj; Nieuwenhuis, Robert; Veith, Helmut; Voronkov, Andrei

Full Paper: http://drops.dagstuhl.de/opus/volltexte/2010/2736

Decision Procedures in Hardware Design

Armin Biere (University of Linz, AT)

In this talk we present an overview on the usage of decision procedures in hardware verification. We start with constrained random simulation, which uses constraint solvers for generating test input. Another lightweight usage of decision procedure appears in the context of semi-formal techniques and bounded model checking. The most successful application of formal techniques in hardware verification is equivalence checking.

Model checking of application specific properties is harder to adopt, but is used in industry.

There are some applications, such as processor verification, where inductive techniques and theorem proving play an important role. Automated combinational and sequential techniques are at the border line of becoming practical.

The talk ends with an overview on some recent results on blocked clauses elimination in SAT.

See also:
http://fmv.jku.at/biere/talks/Biere-Dagstuhl10-talk.pdf

The OpenSMT Solver

Roberto Bruttomesso (Universität Lugano, CH)

Abstract: In this talk we describe OpenSMT, an incremental, efficient, and open-source SMT-Solver.

OpenSMT has been specifically designed to be easily extended with new theory-solvers, in order to be accessible for non-experts for the development of customized algorithms. We sketch the solver's architecture and interface, and we discuss its distinguishing features w.r.t. other state-of-the-art solvers. In particular we describe a preprocessing technique based on the Davis-Putnam and the Fourier-Motzkin methods and we present an efficient decision procedure for a theory of bit-vectors.

Keywords: SMT, bit-vectors, Fourier-Motzkin

Instantiation-Based Interpolation for Quantified Formulae

Juergen Christ (Universität Freiburg, DE)

Interpolation has proven highly effective in program analysis and verification, e. g., to derive invariants or new abstractions. While interpolation for quantifier free formulae is understood quite well, it turns out to be challenging in the presence of quantifiers.

We present in this talk modifications to instantiation based SMT-solvers and to McMillan's interpolation algorithm in order to compute quantified interpolants.

Keywords: Interpolation Quantifier SMT

Joint work of: Christ, Juergen; Hoenicke, Jochen

Full Paper: http://drops.dagstuhl.de/opus/volltexte/2010/2735

Computing Abstractions with SMT solvers

Alessandro Cimatti (Fondazione Bruno Kessler - Trento, IT)

The precise computation of abstractions is a bottleneck in many approaches to CEGAR-based verification. In this paper, we propose a novel approach, based on the use of structural information.

Rather than computing the abstraction as a single, monolithic quantification, we provide a structure-aware abstraction algorithm, based on two complementary steps. The first, high-level step exploits the structure of the system, and partitions the abstraction problem into the combination of several smaller abstraction problems. This is represented as a formula with quantifiers. The second, low-level step exploits the structure of the formula, in particular the occurrence of variables within the quantifiers, and applies a set of low-level rewriting rules aiming at further reducing the scope of quantifiers.

We experimentally evaluate the approach on a substantial set of benchmarks, and show significant speed ups compared to monolithic abstraction algorithms.

Keywords: SMT, Abstraction

Decidable fragments of first-order logic, and combinations

Pascal Fontaine (INRIA - Nancy, FR)

Combination of decision procedures usually consider particular theories like uninterpreted functions, lists, arrays, fragments of arithmetic, bit-vectors. In this talk, we consider some of the best-known decidable fragments of first-order logic with equality.

All theories in the guarded fragment, the loosely guarded fragment, and the packed guarded fragment, are shiny (and thus also stably infinite). It is therefore possible to design a decision procedure for the combination of such theories with an other arbitrary decidable theory.

The Löwenheim class (monadic FOL with equality, but without functions), Bernays-Schönfinkel-Ramsey theories (finite sets of formulas of the form $\exists^*\forall^*\varphi$, where φ is a function-free and quantifier-free FOL formula) and the two-variable fragment of FOL, contain non shiny theories, and even non stably infinite ones. Nevertheless, combination of such theories with an other decidable theory (with minor restriction) is also decidable.

Keywords: Extending decision procedures, combination of decision procedures, first-order decidable classes

Solvers for String Theories

Vijay Ganesh (MIT - Cambridge, US)

JavaScript and PHP-based string-manipulating programs are everywhere on the internet. These programs are plagued by security vulnerabilities. Hence, there is an acute need for SMT-based automated tools that are aimed at bugfinding and analyzing string-manipulating programs.

To address the above problem, we have developed two string solvers, Hampi and Kaluza, for rich theories of strings. I will present these tools, their input languages, the solving algorithm and experimental evaluation.

Keywords: String solvers, PHP, JavaScript, string-manipulating programs

HAMPI: A Solver for String Constraints

Vijay Ganesh (MIT - Cambridge, US)

Many automatic testing, analysis, and verification techniques for programs can be effectively reduced to a constraint-generation phase followed by a constraint-solving phase. This separation of concerns often leads to more effective and maintainable tools. The increasing efficiency of off-the-shelf constraint solvers makes this approach even more compelling. However, there are few effective

and sufficiently expressive off-the-shelf solvers for string constraints generated by analysis techniques for string-manipulating programs.

We designed and implemented Hampi, a solver for string constraints over fixed-size string variables. Hampi constraints express membership in regular languages and fixed-size context-free languages. Hampi constraints may contain context-free-language definitions, regular-language definitions and operations, and the membership predicate. Given a set of constraints, Hampi outputs a string that satisfies all the constraints, or reports that the constraints are unsatisfiable.

Hampi is expressive and efficient, and can be successfully applied to testing and analysis of real programs. Our experiments use Hampi in: static and dynamic analyses for finding SQL injection vulnerabilities in Web applications; automated bug finding in C programs using systematic testing; and compare Hampi with another string solver. Hampi's source code, documentation, and the experimental data are available at `http://people.csail.mit.edu/akiezun/hampi`.

Keywords: SMT String Solvers, PHP and JavaScript bugfinding, SQL Injection and XSS attacks, concolic testing

Full Paper:
http://people.csail.mit.edu/vganesh

See also: @inproceedingsDBLP:conf/issta/KiezunGGHE09, author = Adam Kiezun and Vijay Ganesh and Philip J. Guo and Pieter Hooimeijer and Michael D. Ernst, title = HAMPI: a solver for string constraints, booktitle = ISSTA, year = 2009, pages = 105-116, ee = http://doi.acm.org/10.1145/1572272.1572286, crossref = DBLP:conf/issta/2009, bibsource = DBLP, http://dblp.uni-trier.de @proceedingsDBLP:conf/issta/2009, editor = Gregg Rothermel and Laura K. Dillon, title = Proceedings of the Eighteenth International Symposium on Software Testing and Analysis, ISSTA 2009, Chicago, IL, USA, July 19-23, 2009, booktitle = ISSTA, publisher = ACM, year = 2009, isbn = 978-1-60558-338-9, bibsource = DBLP, http://dblp.uni-trier.de

The Model Checker MCMT for Array Based Systems

Silvio Ghilardi (Università di Milano, IT)

The notion of an array based system has been introduced in [2].

Such systems are declarative abstractions of several classes of parametrised systems and (sequential) programs manipulating arrays. In array based systems, safety properties of infinite state systems can be proved by combining the classical algebraic approach of [1] with deductive techniques exploiting, off-the-shelf, SMT solvers. In fact, a suitable format for initial/unsafe sets of states states as well as for transition formulae can be designed in such a way that: (i) the format is sufficiently expressive to cover interesting classes of benchmarks; (ii) proof obligations arising during backward search analysis can be discharged by instantiation and SMT solving techniques for quantifier-free formulae.

To make the theoretical framework useful in practice, powerful heuristics are required to obtain adequate performances: these heuristics concern optimization of the computation of the pre-image [6], (static and dynamic) filtration of the quantifiers instantiations, as well as forward and backward simplification routines [4]. Last but not least, suitable strategies for invariant generation [3] and predicate abstraction are needed to speed up search and to block trivial non termination sources.

Besides theoretical foundations, we discuss in this talk the functionalities available in our tool MCMT [7,5], in the current release (v. 1.0.1), as well as its performances on a series of classical (but heterogeneous) benchmarks. MCMT is based on the SMT-solver YICES [8].

In the final part of talk more recent case studies involving parametrized timed systems (joint work with A. Carioni) as well as fault tolerant protocols (joint work with F. Alberti, E. Pagani, G.P. Rossi) will be reported.

Keywords: Satisfiability Modulo Theories, Infinite State Model Checking

Joint work of: Ghilardi, Silvio; Ranise, Silvio

Software Model Checking via Large-Block Encoding

Alberto Griggio (Fondazione Bruno Kessler - Trento, IT)

Several successful approaches to software verification are based on the construction and analysis of an abstract reachability tree (ART).

The ART represents unwindings of the control-flow graph of the program.

Traditionally, a transition of the ART represents a single block of the program, and therefore, we call this approach single-block encoding (SBE). SBE may result in a huge number of program paths to be explored, which constitutes a fundamental source of inefficiency. We propose a generalization of the approach, in which transitions of the ART represent larger portions of the program; we call this approach large-block encoding (LBE). LBE may reduce the number of paths to be explored up to exponentially. Within this framework, we also investigate symbolic representations: for representing abstract states, in addition to conjunctions as used in SBE, we investigate the use of arbitrary Boolean formulas; for computing abstract-successor states, in addition to Cartesian predicate abstraction as used in SBE, we investigate the use of Boolean predicate abstraction. The new encoding leverages the efficiency of state-of-the-art SMT solvers, which can symbolically compute abstract large-block successors.

Our experiments on benchmark C programs show that the large-block encoding outperforms the single-block encoding.

Joint work of: Beyer, Dirk; Cimatti, Alessandro; Griggio, Alberto; Keremoglu, Erkan; Sebastiani, Roberto

Symbol Elimination and Interpolation in Vampire

Krystof Hoder (University of Manchester, GB)

It has recently been shown that proofs in which some symbols are colored (e.g. local or split proofs and symbol-eliminating proofs) can be used for a number of applications, such as invariant generation and computing interpolants.

This talk describes how such proofs and interpolant generation are implemented in the first-order theorem prover Vampire.

Joint work of: Hoder, Krystof; Kovacs, Laura; Voronkov, Andrei

Interpolation for Uninterpreted Functions and Linear Arithmetic

Jochen Hoenicke (Universität Freiburg, DE)

For single theories the McMillan algorithm can be used for interpolation in the DPLL(T) framework. However, in theory combination the problem is to find shared terms for variables shared between the theories. Current methods need to reorder the proof to separate the reasoning of the theories. We present a method that works directly on the proof without any reordering.

Keywords: Interpolation, Nelson-Oppen, Delayed Theory Combination, Linear Arithmetic, Uninterpreted Functions

Hierarchical Reasoning: Improving Efficiency and Ensuring Locality

Swen Jacobs (EPFL - Lausanne, CH)

Hierarchical reasoning in local theory extensions allows us decide a certain class of universally quantified SMT problems by reduction to a decidable base theory. The quantified formulas can be seen as an axiomatization of one or several new function symbols. Among other application areas, this allows us to verify safety properties of systems with a parametric number of infinite-state components, if their properties can be expressed in our framework.

In this talk, I will first present an incremental method for the reduction to the base theory, based on the first-order instance generation method of Ganzinger and Korovin. I will show that this improves efficiency for a significant number of examples, while preserving soundness and completeness.

Furthermore, I will present some recent work, first identifying the class of "one-variable axioms" that are local extensions under certain conditions. Then, I introduce "locality-ensuring constraints", which can be used to incorporate these conditions into the axioms for a given background theory. Finally, I show that for the class of one-variable axioms, locality-ensuring constraints can efficiently be computed for some background theories.

Interpolation and Symbol Elimination

Laura Kovacs (TU Wien, AT)

We present several results related to local proofs, interpolation and superposition calculus and discuss their use in predicate abstraction and invariant generation. Our results suggest that symbol-eliminating inferences may be an interesting alternative to interpolation.

Keywords: Interpolation, symbol elimination, Vampire

Joint work of: Hoder, Krystof; Kovacs, Laura; Voronkov, Andrei

Decision Procedures for Security-by-Contract on Mobile devices

Fabio Massacci (University of Trento - Povo, IT)

Security-by-Contract: using automata modulo theory (and decision procedures) to decide on the spot whether a mobile (.NET) device should download an applet.

This is widely different from the usual DP-for-verification talk in which people gives benchmarks on huge machines on how they could verify this or that (or the toy talks of Rajeev Gorè on how to run tableaux on a smart-card).

It is really using an essential NuSMV core DP+emptyness check on the fly on a mobile phone before running your favourite applet and we could even run a nice demo with a couple of phones and a volunteer from the audience.

Theory by F. Massacci and I. Siahaan. Workhorse by Cimatti, Roveri and Tonetta

Quantifier elimination by lazy model enumeration

David Monniaux (VERIMAG - Gières, FR)

We propose a quantifier elimination scheme based on nested lazy model enumeration through SMT-solving, and projections. This scheme may be applied to any logic that fulfills certain conditions; we illustrate it for linear real arithmetic. The quantifier elimination problem for linear real arithmetic is doubly exponential in the worst case, and so is our method. We have implemented it and benchmarked it against other methods from the literature.

Keywords: Quantifier elimination, SMT-solving, linear real arithmetic

Full Paper:
http://www-verimag.imag.fr/~monniaux/biblio/Monniaux_CAV10.pdf

See also: CAV 2010

Verifying Functional Properties with Quantified SMT

Michal Moskal (Microsoft Research - Redmond, US)

VCC is an industrial-strength verification environment for low-level concurrent system code written in C. VCC takes a program (annotated with function contracts, state assertions, and type invariants) and attempts to prove the correctness of these annotations. It includes tools for monitoring proof attempts and constructing partial counterexample executions for failed proofs.

I shortly report on experience of using VCC to verify the Microsoft Hyper-V hypervisor, in particular about the problems with predictibility of quantifier instantiation. I also present sketches of some solutions.

The attached papers were published at SMT2009, TPHOL2009 and will be at CAV2010.

Full Paper:
http://vcc.codeplex.com/

Gröbner Bases and Some Applications in Z3 and RAHD

Grant Olney Passmore (University of Edinburgh, GB)

I'll present the foundations of Abstract Groebner Bases, a theory of Groebner basis procedures Leo de Moura and I have developed for analysing novel Groebner basis construction strategies. This theory is derived from the Bachmair-Dershowitz theory of Abstract Completion and can be seen as a simplification and further development of the Bachmair-Ganzinger view of Buchberger's Algorithm as a constraint-based completion procedure.

An important component of this has been the use of proof-orders (à la Bachmair-Dershowitz, Bachmair-Ganzinger) for proving the strategy-independent admissibility of superfluous S-polynomial criteria which are crucial aspects of efficient Groebner basis construction. We have proven three such criteria to be admissible under Abstract GBs and work continues on more.

I'll then describe how we've used this theory to prove correct novel Groebner basis construction algorithms based on saturation loops used in the ATP community. These procedures have been observed to significantly outperform basis construction algorithms based on Buchberger's algorithm (e.g., BA+Gebaur-Moeller, F4) for L3 ('large, largely linear') nonlinear systems of thousands of equational constraints which arise when SMT solvers are used to verify programs with nonlinear arithmetical components (joint work with Paul B. Jackson). Finally, I'll touch on how these procedures are being used in the SMT solver Z3 and the nonlinear real arithmetic proof procedure RAHD.

Keywords: Groebner bases, nonlinear arithmetic, SMT

Joint work of: Passmore, Grant Olney; de Moura, Leonardo; Jackson, Paul B.

Full Paper: http://drops.dagstuhl.de/opus/volltexte/2010/2734

Complete Functional Synthesis

Ruzica Piskac (EPFL - Lausanne, CH)

Synthesis of program fragments from specifications can make programs easier to write and easier to reason about. To integrate synthesis into programming languages, synthesis algorithms should behave in a predictable way—they should succeed for a well-defined class of specifications. They should also support unbounded data types such as numbers and data structures. We propose to generalize decision procedures into predictable and complete synthesis procedures. Such procedures are guaranteed to find code that satisfies the specification if such code exists. Moreover, we identify conditions under which synthesis will statically decide whether the solution is guaranteed to exist, and whether it is unique. We demonstrate our approach by starting from decision procedures for linear arithmetic and data structures and transforming them into synthesis procedures. We establish results on the size and the efficiency of the synthesized code. We show that such procedures are useful as a language extension with implicit value definitions, and we show how to extend a compiler to support such definitions. Our constructs provide the benefits of synthesis to programmers, without requiring them to learn new concepts or give up a deterministic execution model.

Reasoning about Collections with Cardinality Constraints

Ruzica Piskac (EPFL - Lausanne, CH)

Applications in software verification and interactive theorem proving often involve reasoning about sets of objects. Cardinality constraints on such collections also arise in these scenarios. Multisets arise for analogous reasons as sets: abstracting the content of linked data structure with duplicate elements leads to multisets. Interactive theorem provers such as Isabelle specify theories of multisets and prove a number of theorems about them to enable their use in interactive verification. However, the decidability and complexity of constraints on multisets is much less understood than for constraints on sets.

In this talk we will show that the satisfiability problem for a logic of sets, multisets (bags), and cardinality constraints is decidable. We will also show that the optimal complexity results. It relies on an extension of integer linear arithmetic with a "star" operator, which takes closure under vector addition of the solution set of a linear arithmetic formula. We show that the satisfiability problem for this extended language remains in NP (and therefore NP-complete). Our proof uses semilinear set characterization of solutions of integer linear arithmetic formulas, as well as a generalization of a recent result on sparse solutions of integer linear programming problems.

We also mention several extensions of these results: function image operators, collections with fractional multiplicity, and a result on combination of

non-disjoint theories which share set symbols and operations. In addition to verification, we also report on the use of these decision procedures in synthesis.

Craig Interpolation for Quantifier-Free Presburger Arithmetic

Philipp Ruemmer (University of Oxford, GB)

Craig interpolation has become a versatile tool in formal verification, for instance to generate intermediate assertions for safety analysis. Interpolants are typically determined by annotating the steps of an unsatisfiability proof with partial interpolants. In this work, we consider Craig interpolation for full quantifier-free Presburger arithmetic (QFPA), for which until recently no efficient interpolation procedures were known. Closing this gap, we introduce an interpolating sequent calculus for QFPA and prove it to be sound and complete. In particular, the complexity of extracting interpolants from proofs is discussed. The calculus is flexible and can be enriched with uninterpreted predicates, and thus extended to important theories such as arrays and uninterpreted functions.

Keywords: Craig interpolation, Presburger arithmetic, integer arithmetic, sequent calculus

Joint work of: Brillout, Angelo; Rümmer, Philipp; Kroening, Daniel; Wahl, Thomas

Full Paper:
http://philipp.ruemmer.org/publications/iprincess10.pdf

See also: International Joint Conference on Automated Reasoning (IJCAR) at FLoC, Edinburgh, Scotland, 2010; To appear in LNCS

Variable Dependencies of Quantified CSPs

Marko Samer (TU Wien, AT)

Quantified constraint satisfaction extends classical constraint satisfaction by a linear order of the variables and an associated existential or universal quantifier to each variable. In general, the semantics of the quantifiers does not allow to change the linear order of the variables arbitrarily without affecting the truth value of the instance. In this paper we investigate variable dependencies that are caused by the influence of the relative order between these variables on the truth value of the instance. Several approaches have been proposed in the literature for identifying such dependencies in the context of quantified Boolean formulas. We generalize these ideas to quantified constraint satisfaction and present new concepts that allow a refined analysis.

Full Paper:
http://dx.doi.org/10.1007/978-3-540-89439-1_36

The Synergy of Precise and Fast Abstractions for Program Verification

Natasha Sharygina (Universität Lugano, CH)

Predicate abstraction is a powerful technique to reduce the state space of a program to a finite and affordable number of states. It produces a conservative over-approximation where concrete states are grouped together according to a given set of predicates. A precise abstraction contains the minimal set of transitions with regards to the predicates, but as a result is computationally expensive. Most model checkers therefore approximate the abstraction to alleviate the computation of the abstract system by trading off precision with cost. However, approximation results in a higher number of refinement iterations, since it can produce more false counterexamples than its precise counterpart. The refinement loop can become prohibitively expensive for large programs.

In this talk I will present a new abstraction refinement technique that combines slow and precise predicate abstraction techniques with fast and imprecise ones. It allows computing the abstraction quickly, but keeps it precise enough to avoid too many refinement iterations.

We implemented the new algorithm in a state-of-the-art software model checker. Our tests with various real life benchmarks show that the new approach systematically outperforms both precise and imprecise techniques.

Keywords: Program model checking, Abstraction

Full Paper:
http://www.verify.inf.unisi.ch/projects/synergy

Hierarchical reasoning for the verification of parametric systems

Viorica Sofronie-Stokkermans (MPI für Informatik - Saarbrücken, DE)

We will present certain classes of verification problems for parametric reactive and hybrid systems, and identify the types of logical theories which can be used for modeling such systems and the reasoning tasks which need to be solved in this context:

The first problem we address is to check whether a safety property, expressed by a suitable formula, is an invariant, or holds for paths of bounded length, for given instances of the parameters, or under given constraints on parameters.

For this type of problems, we aim at identifying situations in which decision procedures exist. We show that this is often the case, by investigating consequences of locality phenomena in verification. If unsafety is detected, the method we use allows us to generate counterexamples to safety, i.e. concrete system descriptions which satisfy all the imposed constraints and are unsafe.

We also analyze the dual problem, related to system synthesis, of deriving constraints between parameters which guarantee that a certain safety property is an invariant of the system or holds for paths of bounded length.

We analyze situations in which the parameters can be constants or functions (with certain properties). We present a new approach and show that sound and complete hierarchical reduction for SMT checking in local extensions allows to reduce the problem of checking that certain formulae are invariants to testing the satisfiability of certain formulae. Quantifier elimination is used for generating constraints on the parameters of the system (be they data or functions) which guarantee safety. These constraints on the parameters may also be used to solve optimization problems (maximize/minimize some of the parameters) such that safety is guaranteed. If we also express invariants in a parametric form, this method can also be used for identifying conditions which guarantee that formulae with a certain shape are invariants, and ultimately for generating invariants with a certain shape.

We illustrate the methods we use on several examples.

Conflict Resolution

Nestan Tsiskaridze (University of Manchester, GB)

We presented a new method for solving systems of linear constraints over the reals - conflict resolution method. The method works with a set of constraints and an assignment on variables. It iteratively refines the assignment trying to make it into a solution. If such a refinement is impossible, it is due to a pair of conflicting constraints. We resolve a conflict by deriving a new constraint and adding it to the system. The refinement is done successively until either the assignment becomes a solution of the system or the inconsistency of the initial system is proved. Generation of new constraints ensures that new constraints are non-redundant in some natural sense. The method also has a number of properties crucial for integration into SMT solving: incrementality and ability to generate explanations for an unsatisfiable subset of constraints. Our method can easily be made incremental: after adding/removing constraints we can always continue with the current assignment. Explanations can be generated from the proofs of unsatisfiability which are easily extractable from runs of the conflict resolution method. The method may also have an independent value for linear optimisation and a number of other combinatorial problems. We implemented a solver based on our method. The solver showed itself to be highly competitive with existing state-of-the-art solvers and in some cases even outperforming them.

Keywords: Satisfiability Modulo Theories, SMT, Linear real and integer arithmetic, Linear Programming

Joint work of: Korovin, Konstantin; Tsiskaridze, Nestan; Voronkov, Andrei

Full Paper:
http://www.springerlink.com/content/d3117048176jq0l5/

See also: K. Korovin, N. Tsiskaridze, A. Voronkov. Conflict Resolution. CP 2009, LNCS v. 5732, Springer Verlag 2009, pp. 509-523.

Decision Procedures for Data Structures

Thomas Wies (IST Austria - Klosterneuburg, AT)

Motivated by the problem of deciding verification conditions for the verification of functional and imperative programs, we present new decision procedures for automated reasoning about data structures.

We first explore automated reasoning about the non-disjoint combination of theories that share set variables and set operations. We present a combination theorem that ensures decidability of the set-sharing combination of theories that are reducible to the theory of sets with linear cardinality constraints (BAPA).

We then apply our combination theorem to the special case of automated reasoning about functional lists. We show how to decide in NP the satisfiability problem for logical constraints containing equality, list constructors, selectors, as well as the transitive sublist relation. We then extend this class of constraints with operators to compute the set of all sublists, and the set of objects stored in a list. Finally, we support constraints on sizes of sets, which gives us the ability to compute list length as well as the number of distinct list elements. We show that the extended theory is BAPA-reducible, and therefore still in NP. This reduction enables us to combine our theory with other decidable theories that impose constraints on sets of objects, which further increases the potential of our decidability result in software verification.

Keywords: Automated reasoning, software verification, non-disjoint combination of theories, local theory extensions

Forward Analysis of Depth-Bounded Processes

Thomas Wies (IST Austria - Klosterneuburg, AT)

We study the verification of message passing systems that admit unbounded creation of threads and name mobility. Depth-bounded processes form the most expressive known fragment of such systems for which interesting verification problems are still decidable. A configuration of a message passing system can be represented as a graph. In a depth-bounded system the lengths of the acyclic paths in all reachable configurations are bounded by a constant. Many real-life use cases of message passing concurrency are depth-bounded. We develop an adequate domain of limits for the well-structured transition systems that are induced by depth-bounded processes. An immediate consequence of our result is

the existence of a forward algorithm that decides the covering problem for this class. Unlike backward algorithms, the forward algorithm terminates even if the depth of the system is not known a priori. More importantly, our result suggests a whole spectrum of forward algorithms that enable the effective verification of a large class of mobile systems and concurrent programs based on message passing.

Keywords: Verification, program analysis, message passing concurrency, well-structured transition systems

Joint work of: Wies, Thomas; Zufferey, Damien; Henzinger, Thomas A.

References

1. P. A. Abdulla, K. Cerans, B. Jonsson, and Y.-K. Tsay. General decidability theorems for infinite-state systems. In *Proc. of LICS*, pages 313–321, 1996.
2. S. Ghilardi, E. Nicolini, S. Ranise, and D. Zucchelli. Towards SMT Model-Checking of Array-based Systems. In *Proc. of IJCAR 08*, LNCS, pages 67–82, 2008. Full version available as a Technical Report at http://homes.dsi.unimi.it/~ghilardi/allegati/GhiNiRaZu-RI318-08.pdf.
3. S. Ghilardi and S. Ranise. Goal Directed Invariant Synthesis for Model Checking Modulo Theories. In *Tableaux 09*, LNAI, pages 173–188. Springer, 2009.
4. S. Ghilardi and S. Ranise. Model Checking Modulo Theory at work: the intergration of Yices in MCMT. In *AFM 09 (co-located with CAV09)*, 2009.
5. S. Ghilardi and S. Ranise. MCMT: A Model Checker Modulo Theories (system description). In *Proc. of IJCAR 10*, LNCS, pages 22–29, 2010.
6. S. Ghilardi, S. Ranise, and T. Valsecchi. Light-Weight SMT-based Model-Checking. In *Proc. of AVOCS 07-08*, ENTCS, 2008.
7. MCMT. http://homes.dsi.unimi.it/~ghilardi/mcmt.
8. Yices. http://yices.csl.sri.com/.